Introduction to Computer Organization – Fall 2023

Homework 4

Due: @11:55pm, Monday December 4th on Gradescope

Name: ______________________________ Uniqname: __________________________

1. Submit a pdf of your typed or handwritten homework on Gradescope.

2. Your answers should be neat, clearly marked, and concise. Typed work is recommended, but not required unless otherwise stated. Show all your work where requested, and state any special or non-obvious assumptions you make.

3. You may discuss your solution methods with other students, but the solutions you submit must be your own.

4. **Late Homework Policy:** Submissions turned in by 1:00 am the next day will be accepted but with a 5% penalty. Assignments turned in between 1:00 am and 11:55 pm will get a 30% penalty, and any submissions made after this time will not be accepted.

5. When submitting your answers to Gradescope you need to indicate what page(s) each problem is on to receive credit. The grader may choose not to grade the homework if answer locations are not indicated.

6. **The last two questions are group questions**.
   - **You will turn those questions in separately** and may do it in a group of up to two students (yes, you can do it by yourself if you wish).
   - It is an honor code violation if a student is listed as contributing who did not actually participate in working on that problem.
   - Further, we suggest that you not split this up but rather work on the problem as a group.
7. After each question (or in some cases question part), we've indicated which lecture number we expect to cover the relevant material. So "**(L7)**" indicates that we expect to cover the material in lecture 7. **Note, that two of the problems won't be doable until the Thursday before the homework is due.**

Problem 1: Out of Control [16 points]

For the questions below, reference the following piece of LC2K assembly code: **(L15)**

```
      lw    0     1     pos1  1                      reg1: 1
      lw    0     2     neg1  -1                     reg2: -1
      lw    0     3     count  5                     reg3: 5
      nor   1     1     4
loop  beq   0     3     fin         //beq0
      nor   3     3     5          →        compare reg3 with 0.
      nor   4     5     5
      beq   0     5     even        //beq1
      add   6     1     6
      beq   0     0     next        //beq2
even  add   7     1     7
next  add   3     2     3
      beq   0     0     loop        //beq3
fin   halt
count .fill 5
pos1  .fill 1
neg1  .fill -1
```

a) Write the sequence of branch decisions for each beq instruction. Let “taken” be denoted as “T” and “not taken” as “N”. For example, a beq that is taken twice and then not taken once would have a sequence TTN. **[4]**

beq0	
beq1	
beq2	
beq3	

b) How many extra cycles are added due to the following schemes by squashes, assuming the 5-stage pipeline from lecture? Show your work.

i) Speculate and Squash: Predict always taken **[3]**

ii) Speculate and Squash: Predict backwards taken, forwards not taken **[3]**

iii) Local 1-bit Branch Predictor: Initialized to taken **[3]**

iv) Local 2-bit Branch Predictor: Initialized to strongly taken **[3]**

$4GB \rightarrow 2^2 \cdot 2^{30} = 2^{32}$

Problem 2: Cache Overhead [15 points]

A new company has proposed a number of different cache layouts for their system and you've been asked to come in and calculate the overhead for each of the different caches. Their system uses a cache with 32KB of data storage capable of addressing 4 GB of byte-addressable memory. Stores will be handled by write-back and allocate-on-write policies. *Please be sure to show the work for your calculations.* **(L19)**

A. The first design is a fully-associative cache with a block size of 32 bytes, **how many bytes of overhead would the cache have in total** (including any necessary tag bits, valid bits, dirty bits, or LRU bits)? **[5]**

32KB 32B K: 2^{10} line.

| tag | 32B |

2^5

$32-5=27$

27 +1 +1

B. Their next design utilizes a direct-mapped cache with 256 different cache lines. **How many bytes of overhead would the cache have in total** (including any necessary tag bits, valid bits, dirty bits, or LRU bits)? **[5]**

32KB

$256 \rightarrow \log_2(256) = ~~2~~8$

2^8

$$\frac{2^5 \times 2^{10} B}{2^8}$$

$$\frac{2^{15}}{2^8} B = 2^7 B.$$

17 × 256.

| tag | line index. | offset |

17 8

32-7=25. 27

~~8bit LRU~~

25 tag.

1 bit dirty

1 bit valid.

8+2+25=35

17+1+1=19

C. Finally, they've suggested a 2-way set associative cache with 256 sets. **How many bytes of overhead would the cache keep in total** (including any necessary tag bits, valid bits, dirty bits, or LRU bits)? **[5]**

$256 = 2^8$

2^8 set.

$$\frac{32KB}{512} \quad \frac{2^5 \times 2^{10}}{2^9} = 2^6$$

512 line

Valid : 1

dirty : 1

LRU : 1

| tag | set index | offset |

18 8 6 bit.

$32-8-6 = 32-14 = 18$

tag : (18 + 1 + 1 + 1) × 512

32 Byte cache.

Set 0 0000

$32 \div 8 = 4$

8 bit

Tag | Set index | offset

1 2^3

Set 1 0110 ~~0100~~ 1111

Problem 3: Classifying Misses [18 points]

Consider a 2-way set-associative 32-byte cache for an 8-bit system. Each block is 8 bytes and an LRU eviction policy is used. Given the following sequence of cache accesses, determine whether each access is a hit or a miss and then classify each of the misses as one of compulsory, capacity, or conflict.

Additionally, break down each address to show the tag in order to help see what is in the cache. Please follow the format of the first row (which is filled out for you). (L20)

0

E 6 4 7

12

0000

Address	Tag (binary)	Index	Hit/Miss	Compulsory	Capacity	Conflict
0x01 0001	0000	0	Miss	X		
0x05 0101	0000	0	Hit			
0x4C 1010	0100	1	Miss	X		
0x47 0111	0100	0	Hit			
0x07 0111	0000	0	Hit			
0xFC 1100	1111	1	Miss	X		
0x69 1001	0110	1	Miss	X		
0x06 0110	0000	0	Hit			
0x48 1000	0100	1	Miss		X	~~X~~
0xEA 1010	1110	1	Miss	X		
0xFC 1100	1111	1	Miss		X	
0x03 0011	0000	0	Hit			

Problem 4 – A real page turner [15 points]

Consider a 40-bit, byte-addressable system that uses virtual memory. The system has a maximum of 128 GB of physical memory installed with a page size of 4 KB. **(L23)**

a) How many bits are used for the page offset? How many bits are used to index into the page table(s)? **[3]**

Page Offset: _________ Virtual Page Number Bits: _________

b) How many **virtual** pages exist in the system? **[3]**

c) How many **physica**l pages exist in the system? **[3]**

d) If the system has a single-level page table with 4 byte entries, how large must the page table be (in bytes) to map all the system's virtual memory? **[3]**

e) If we switch to a hierarchical page table such that each page table must fit in a single page, how many page table levels would we need to map all of the system's virtual memory? Assume page table entries are still 4 bytes each. **[3]**

Problem 5: Class evaluations and exit surveys [8 points]

Take a screenshot that shows you did at least two end-of-term course evaluations for EECS 370 one for the lecture and one for your lab leader. We really do read all of these and that can be very useful in helping with future terms. We do realize that because of the way 370 works, you may end up not being able to do an evaluation for the lecture you actually watched/attended. **(N/A)**

Problem 6: I’m having cache flow problems [14 points, group]

Consider the following access pattern: A, B, C, A. Assume that A, B, and C are memory addresses each of which are in a different block of memory. Further, assume A, B and C are generated in a uniformly random way (each block is equally likely) and that a true LRU replacement algorithm is used. *Show your work.* **(L19)**

What is the probability that the second instance of "A" will be a hit if:

i. The cache has 2 lines and is fully-associative **[2]**.

ii. The cache has 4 lines and is fully-associative **[2]**.

iii. The cache has 2 lines and is direct-mapped **[2]**.

iv. The cache has 4 lines and is direct-mapped **[2]**

v. The cache has 4 lines and is two-way set associative? **[4]**

vi. Redo part iii for the following pattern “A, B, C, B, A”. **[2]**

Problem 7: Two-level page table [14 points, group]

You are checking the correctness of a system with virtual memory and no caches. The system uses a 2-level page table scheme with the following partitioning for virtual addresses: **(L23)**

L1 Page Table Index	L2 Page Table Index	Page Offset
Bits 11-8	Bits 7-4	Bits 3 - 0

The page table translates virtual addresses into physical addresses of the following form:

Physical Page Number	Page Offset
Bits 10 - 4	Bits 3 - 0

Each page table entry is **1 byte** wide, and it is stored in (physical) memory. The format of each page table entry is as follows:

Valid bit	Page Number
Bit 7	Bits 6 - 0

A snippet of the machine's physical memory is given in the next page (contents are laid out 16 bytes per row, the first entry being the lowest-order byte. For example, address 0x1 contains the byte 0xf2). You also know that the page table **base register** points to physical address **0x10**.

Translate the two virtual addresses in the next page into physical addresses. Specify the address (not the index!) of the L1 page table entry of interest and its value, and do the same for second-level page tables. Write "–" to indicate an invalid or unknown entry/address. [Hint: You need to use page table entry size and page table index to determine the location of a page table entry in memory.]

Virtual Address	Address of L1 Page Table Entry	Content of L1 Page Table Entry	Address of L2 Page Table Entry	Content of L2 Page Table Entry	Physical Address	Content of Physical Address
0x055						
0x12d						

Snippet of physical memory (All values in hexadecimal):

Address	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f
0x0	f0	f2	00	00	00	00	00	00	00	a4	40	23	44	12	12	12
0x10	8f	8c	c9	4f	00	3f	42	00	00	00	8a	a3	9f	91	00	19
0x20	00	00	00	00	00	12	00	00	00	00	00	00	00	00	00	00
0x30	00	00	00	00	00	00	19	00	00	00	00	00	00	56	00	00
0x40	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x60	80	99	38	00	8f	90	91	92	c4	d3	00	00	34	00	00	00
0x70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0x90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0xa0	00	b8	00	ba	00	b9	00	00	00	40	00	00	00	00	00	00
0xb0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0xc0	00	00	83	b4	b5	00	b6	00	00	00	00	00	00	00	00	00
0xd0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0xe0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0xf0	12	d1	b3	44	00	86	00	00	00	b4	00	00	00	00	00	00
0x100	00	00	02	00	00	00	00	00	00	00	00	00	00	00	00	00
0x110	00	00	00	00	00	00	a3	a2	a1	c0	00	00	00	00	00	00
0x120	00	00	00	00	00	00	00	00	00	00	44	00	00	00	00	00
0x130	00	34	00	00	55	55	55	67	00	88	24	00	4a	01	00	3c